第1章 仕様検討やアーキテクチャ設計を重視する開発プロセスへシフト

── なぜASIC開発にUMLやSystemCが必要なのか

宮原忠義，萩原 篤，守田直也

ここでは，ASIC開発やシステムLSI開発にUML（unified modeling language），SystemCを導入する必要性について議論する．歴史的に，LSI設計においてチップに付加価値を付ける工程は下流から上流へと移行している．かつてはレイアウト設計や論理回路設計が，現在はおもにRTL（register transfer level）の機能設計が付加価値の源泉となっている．今後は，アーキテクチャ設計，アルゴリズム設計，仕様設計がASIC設計者の重要なしごとになる．一方，組み込みソフトウェア開発との並行開発や協調作業も求められている．そのため，ASIC開発においても，ソフトウェア開発と同じように要求分析やテスト仕様設計が重要になる．こうした部分を強化していかないと，ASIC開発やシステムLSI開発の業務そのものを継続できなくなる可能性がある．（編集部）

筆者らが所属する部署では，コピーやプリンタ，ファクシミリなどの複合機に搭載する数百万ゲート規模の制御用ASICや画像処理用ASICを開発しています．ASIC開発におけるプロセス（設計工程）改革を目指した社内ワーキング・グループの結成をきっかけに，筆者らは現在，UMLやSystemCなどの導入・評価を行い，一部ではすでに活用しています．

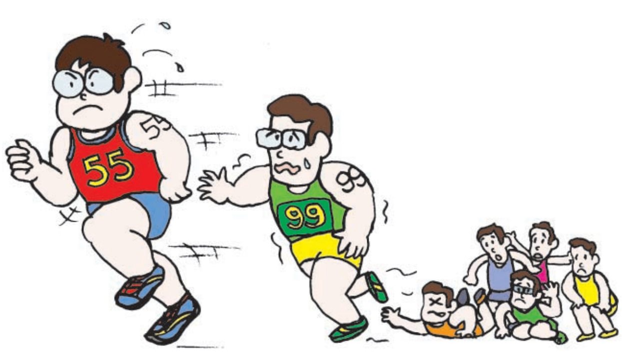


図1　ASIC開発は鉄人レース

タイミング収束，システム・レベル・モデリング，機能検証，低消費電力設計，製造容易化設計（design for manufacturability）など，ASIC開発における課題は年々増えている．新しく登場する手法や考えかたを自分の中で逐次消化していかないと，ASIC開発そのものを断念しなければならなくなるかもしれない．一度，遅れをとると，後から挽回することは難しい．

UMLはオブジェクト指向のソフトウェア開発のために考案された表記法です（例えば，UML 2.0では12種類の図が定義されている）．UMLの規格に合わせて図面を作成し，それに沿ってプログラム（ソース・コード）を開発していきます．SystemCはC++をベースとするシステム・レベル言語です．クラスの機能を用いてC++を拡張し，ハードウェアの動作を記述しやすくしています．

社内ワーキング・グループのメンバは，電気・電子系出身のハードウェア技術者と情報系出身のソフトウェア技術者が半々で構成されており，ハードウェアとソフトウェアの文化や技術を融合しながら活動しています．

● 今傍観していて，果たして後から追いつけるのか？

ASIC設計の手段は，トランジスタ・レベル，回路図，HDL（ハードウェア記述言語）と10年単位で変わっています．現在は，C/C++をベースとした言語が注目を集めています．ほんとうにC/C++全盛の時代がくるのかどうかわかりませんが，徐々にSystemCなどが広がり，EDA（electronic design automation）ツールが整備されてきているのは事実です．読者のみなさんの中にも，時代の変化を予感されている方がいらっしゃるかもしれません．

筆者らの間でも，「設計言語はまだ変わらない」，「もっと広まってから活動しても遅くない」といった声が上がっています．その一方で，最新プロセス（半導体製造技術）やLSI開発手法の変化は速く，「今傍観していて，果たして後

から追いつけるのか？」という不安を抱えています（**図1**）．この不安を払拭するためにも，筆者らは新しい手法を試行して，良いものはどんどん吸収していくことに努めています．

● RTL設計者が要らなくなる!?

あなたが機器メーカのASIC設計者ならば，ASIC開発の5年後，10年後を考えてみてください．果たしてRTL設計の工程を自社で持ち続けていると思いますか？ 仕様を渡して，RTL設計はASICベンダや設計委託会社にお任せ，となっている可能性があることは否定できないと思います．

また，ビヘイビア合成技術が進歩すれば，C言語で書かれたアルゴリズム記述をわざわざVerilog HDLやVHDLに変換する必要がなくなるかもしれません．「将来，RTL設計者は要らなくなるのでは…」，筆者らはこんな危機感を感じ始めています．このような状況では，自社に残すものは適切な機能仕様やアーキテクチャ仕様を模索し，決定する業務，いわゆる上流設計のみになると思われます．少なくとも，仕様やアルゴリズム，アーキテクチャなどを取り扱う工程の比重が高まることはまちがいないでしょう．

LSI設計の技術革新は急速に進んでいます．そのため，ASIC設計者はつねに開発フローの将来像をイメージしておく必要があると考えます．例えば筆者らの場合，オブジェクト指向モデリングやUMLの表記法，SystemCを採用した場合の開発フローを**図2**のようにイメージしています．前述の上流設計は，要求（項目）からアーキテクチャ設計までの工程に相当します．要求分析の工程では，UMLを用いてモデリングを行い，モデルを洗練していくことを思い描いています．

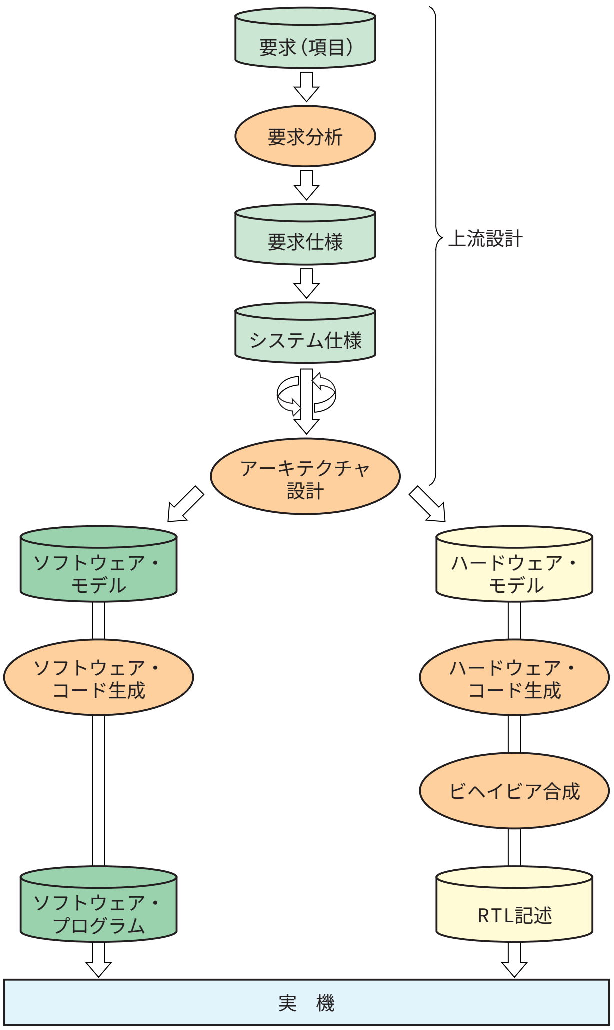


図2　将来の設計フローの例
オブジェクト指向モデリング，UML，SystemCを意識した設計フロー．システム仕様はおもに機能決定と機能分割を，アーキテクチャ設計はハードウェア・ソフトウェア分割，CPU決定などをイメージしている．本設計フローはあくまでも一つの例である．

● UMLは難しいのか？

社内の設計技術者とUMLについて話していると，「UMLは難しい」という答えがよく返ってきます．しかしUMLは単なる表記法であり，UMLを書くことそのものはそれほど難しいはずがありません．おそらく，ほんとうに敷居が高いのは，UMLではなく，要求分析やモデリングの作業なのだと思います（**図3**）．

この敷居を低くするために筆者らが取り組んできたことは，まずは「クリティカル・シンキング」です．新聞の社説を読み，これをフリー・フォーマットの図で表すことを実施しました．なぜ，これが要求分析に生きてくるのかは後で述べます．



図3　UMLは難しくない
UMLは表記法であり，深追いしなければ習得は難しくない．これを使った要求分析やモデリングが難しい．習得にも時間がかかる．

次に，モデリングについては，オブジェクト指向的な思考法を養うために，『思考系UMLモデリング即効エクササイズ』（翔泳社 刊）[1]の問題をひととおり解きました．ひとりではなく数名でいっしょに勉強し，それぞれの回答を持ち寄って討論を行いました．これを繰り返していくうちに討論内容がみるみる変わり，メンバ全員がおもしろいほどオブジェクト指向に目覚めていきました．

将来のためにこのような勉強会を開いて，多くの人に参加してもらいたいのですが，「ASIC開発のしごとで手いっぱいで，参加しているひまがない」，「そう簡単にASIC開発の文化を変えることはできない」などの声も多く，設計技術者を取り巻く環境は厳しいようです．

こういった障壁に対して，実績を重ね，UMLの優位性をもっと広くアピールしていかなければならないと，筆者らは強く感じています．

● 回路設計への本格導入は断念

次に，回路設計，要求管理，検証用のテストベンチ開発においてUMLの導入に取り組んだ事例を紹介します．

設計レビューに参加しているとき，図4のように感じる方も多いかと思います．以下に，回路設計において筆者らが困っていることや改善項目を挙げます．

- 書いたモジュールのソース・コードがスパゲティ構造になっていないか？
- モジュールの役目がひと目でわかる図が欲しい
- ソース・コード・チェック（コード・レビュー）では，図を使ってチェックしたい
- モジュール構成がほんとうに最適かどうかレビューしたい
- 大まかなモジュール構成はドキュメントに残すが，細かいところは各人の頭の中にしかない．これもドキュメント化したい
- 固定/可変を考慮したモジュール構成にしたい
- ASICのモジュール構成や制御手順を，ソフトウェア技術者もハードウェア技術者も共通の図で説明できるようにしたい

これらを解決するため，そして回路の保守性や再利用性を高めることを目指して，オブジェクト指向モデリングを回路設計に適用してみました．また，設計情報の可視化（ドキュメントの充実）を目指して，UMLの導入を試みました．

DMAコントローラのモデリングにUMLを適用した結果については，本誌2004年12月号，pp.88-96の記事「LSIをUMLとSystemCでモデリングする」で述べたので，説明を割愛します．上記のような課題を解消できる期待が持てることは確認できました．



図4　レビュー時のひとコマ

モジュール構成がわかる仕様書のレビューやソース・コードのレビューにおける参加者の思いの一例．これらを解決するのが，UMLやオブジェクト指向によるモデリングだと期待している．

ただし，その後，筆者らは本格導入には至っていません．筆者らは回路仕様をUMLで表記するのではなく，これまでの仕様書フォーマットを改良する方向に向かっています．前述したとおり，ASIC設計者にとってモデリングの敷居が高いことが大きな障壁となっています．モデルを段階的に洗練していくような開発を行わないのであれば，あえて回路設計でUMLを使用する必要はないと筆者らは考えています．

● 要求管理にユースケース図を利用

機能もれや機能の不整合は，ASICの機能仕様書を見てもなかなか発見できない不ぐあいの代表だと思います．このような機能仕様書に混入する不ぐあいは，回路設計や検証を行っている段階では見つけづらく，システムどうしの結合テストを行ったときに初めてわかることがしばしばです．

筆者らは，こうしたトラブルをなくすためにUMLのユースケース図を利用して要求管理を行っています．また，このユースケース図からテスト・シナリオを自動的に生成する手法を試行しています．

どのような製品の開発でも何らかの要求があり，それらの要求を満たすように開発されていると思います．つまり，要求は製品完成時のチェック・リストに相当するものです．しかし，要求仕様書を作り，そこからシステム仕様書や機能仕様書，テスト仕様書を作成するといったことは，かならずしも体系的に行われていないようです．おそらく図5(a)のように，システム設計者や回路設計者，検証エンジニアが頭の中で要求を整理し，そこからシステム仕様書や機能仕様書を作成しているのだと思います．

しかし，増え続ける要求を頭の中だけで管理するのは限界があります．ASICの開発に多くの人々がかかわってくるようになると，さまざまな問題が発生します．関係者の認識を一致させ，要求を満たす製品を開発するには，体系的に要求を管理する図5(b)のようなしくみが必要となります．

要求を体系的に管理するため，筆者らは要求管理チームを作りました．これは，設計者間の認識を一致させるための要求仕様書の作成と保守を目的としたチームです．要求仕様をまとめる際に，最初からすべてが確定することはま

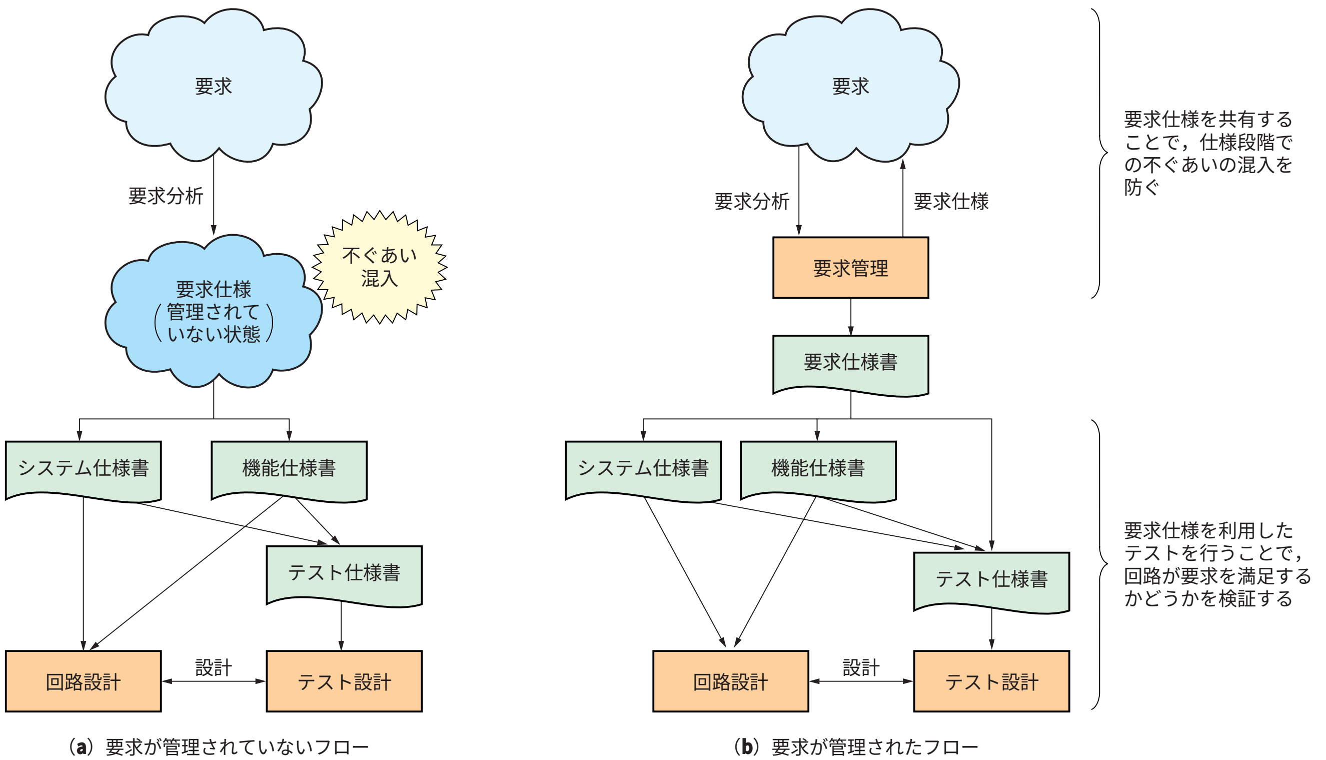


図5　要求管理と設計フロー

(a)のように要求仕様が管理されていない状態で仕様書が作られ，その仕様書に不ぐあいが残っていると，不ぐあいがある状態が正しい状態として設計や検証が行われてしまう．(b)のように要求仕様が管理されている状態で仕様書が作られると，機能仕様書やシステム仕様書を作成する際に残った仕様の不ぐあいをテスト(検証)の段階で発見することができる．



（a）本質的な要求を見つけ出す　（b）だれが見てもわかるものにまとめる

図6　要求管理に必要なスキル

要求管理には，さまざまな形式で書かれた要求から本質的な部分を見つけ出すスキルと，抜き出された仕様を設計者のだれが見てもわかるように要求仕様書としてまとめるスキルが必要になる．

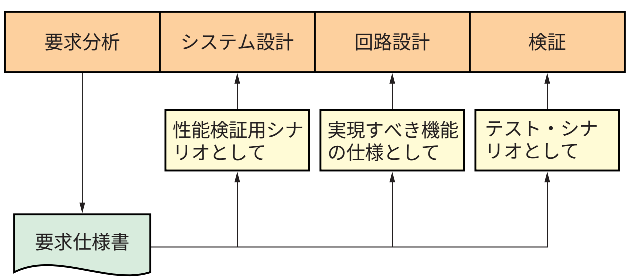


図7　要求仕様書を活用できる局面

要求分析によって作られた要求仕様は，その後の設計フェーズにおいてさまざまな形で利用される．要求仕様書をユースケース図としてまとめておくことで，比較的簡単にそれぞれのフェーズにあった情報を提供できる．

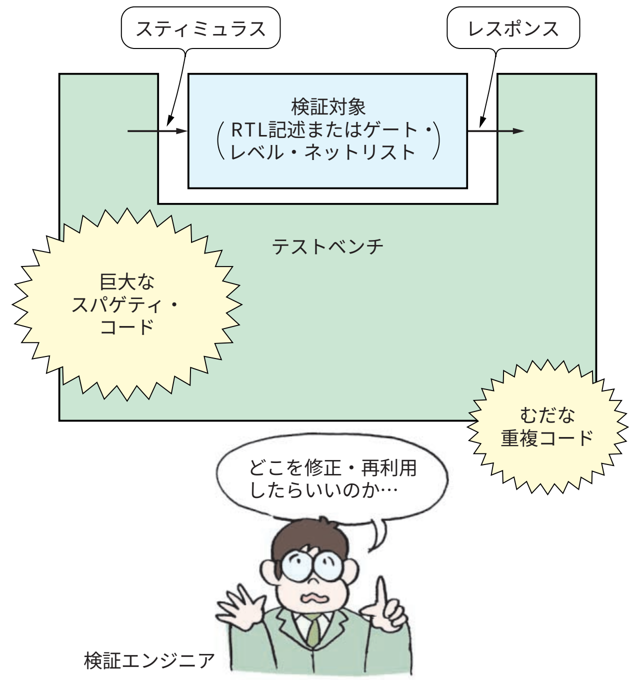


図8　従来のテストベンチ

テストベンチでは，検証対象にスティミュラス（刺激）を入力し，レスポンス（応答）を観測することで動作の検証を行う．テストベンチは巨大化し，かつ複雑な依存関係を持つスパゲティ・コードとなってきている．

れだと思います．要求元がどこなのかを管理し，定期的な要求の確認を行うようにしています．また，現時点でそれぞれの要求仕様がどのような状態にあるのかを管理します．

● ASIC設計には要求分析工程が必要

要求は，その人の立場によりさまざまな視点で表現されます．同じ機能でも，視点が変わるとその表現のしかたは変わります．こうした要求から本質的な部分を抽出し，要求仕様を完成させるためには要求分析の作業が必要となります．要求分析では，図6のように要求の本質的な部分を見つけ出すスキルと，それを要求仕様書としてまとめるスキルが求められます．

要求から本質的な部分を抽出するスキルとして，クリティカル・シンキングで培った能力が役に立ちます．すなわち，文章の中から本質的な部分を読み取り，それを図解する能力です．また，抽出された本質的な要求を要求仕様にまとめる手段として，ユースケース図を利用します．これにより，すべての設計者が共通の認識を持つことができるようにしています．この工程でまとめられた資料を利用してレビューを行い，要求元と設計者の認識を合わせます．

要求仕様がある程度完成すると，それらをもとにシステム（ASIC）の設計が開始されます．設計されたシステムについては，要求を満たすかどうかをチェックする必要があります．要求仕様書からテスト・シナリオを作成し，そのシナリオを流す（実行する）ことによって，要求を満たすかどうかを確認できます．

要求仕様書はユースケース図を利用して書いているため，設計情報と組み合わせることにより，テスト・シナリオの自動生成が可能となっています．

要求仕様は設計者間の認識を一致させることや機能もれ，機能の不整合を防ぐだけでなく，図7のように設計開発のあらゆるフェーズにおけるチェック項目として利用できます．ユースケース図を利用して要求仕様を書くことにより，さまざまな用途で比較的容易に再利用できることがわかりました．今後は，より多くの局面で利用できるように，記述方法を改善していきたいと考えています．

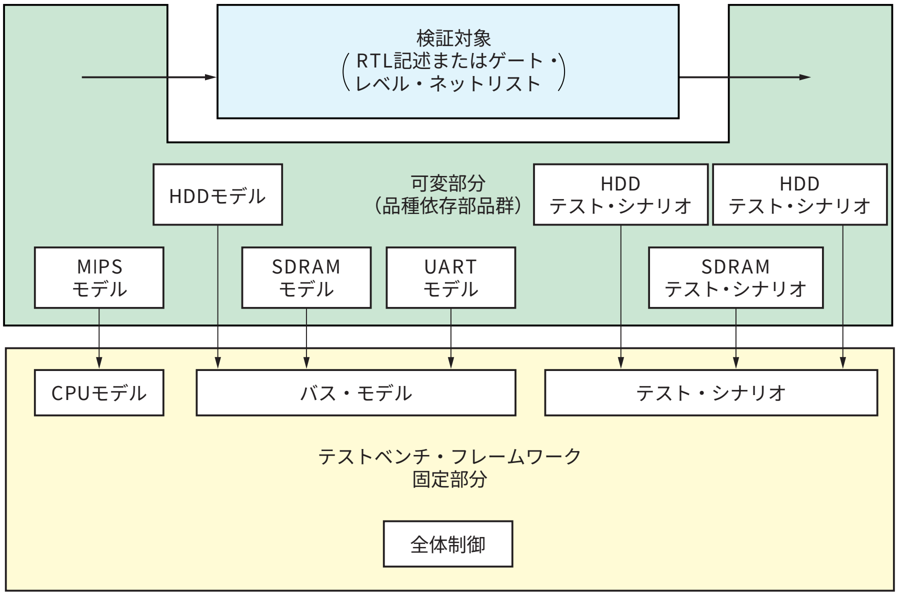


図9
C++テストベンチの全体像
筆者らが構築したテストベンチ・フレームワークは，CPUモデル，バス・モデル，テスト・シナリオ部品を接続する口を提供する．ASICの品種に依存するCPUモデルやバス・モデル，テスト・シナリオをフレームワークとDUT（device under test；検証対象）に接続することで，テストベンチが完成する．

● テストベンチは固定部分と可変部分に分ける

設計するASICのゲート規模の増大に伴って，テストベンチのコード量が増加しています（**図8**）．そのため，従来のテストベンチの構成と運用について，以下の三つの問題が顕在化しています．

- 再利用性や保守性が低い
- コードの可読性が低い
- テストベンチの構築作業を分担しにくい

これらを解決するため，筆者らの検証チームはC++テストベンチを利用しています．

C++はオブジェクト指向に対応しています．オブジェクト指向では，データ（属性）とそのデータに対する処理（ふるまい）をクラスという単位でパッキングします．そのため，テストベンチに変更がある場合は，該当するクラス単位で変更するため，変更箇所が最小限になります．

また，オブジェクト指向をベースとしたUMLの表記法を利用できます．UMLはモデル（設計図）を図や記号で表現するので，設計を手助けしてくれます．UMLの表記法の一つにクラス図があります．クラス図は，クラスの構造とクラス間の関係を表現したものです．クラス図を使うとテストベンチの全体像を表せるので，注目したいクラスが一目瞭然になります．

加えて，SystemCやSCV（SystemC Verification Library；トランザクション形式のデータ記録や制約条件付きランダム・パターン生成に対応した検証用ライブラリ）などの検証環境も整ってきています．

テストベンチは，**図9**のように固定部分と可変部分に分離して作成しています．固定部分はASICの品種に依存しない部分で，どの品種でもそのまま再利用が可能です．検証エンジニアは，可変部分を作り込むことで個々のASICに対応したテストベンチを完成させます．

C++によるテストベンチ開発の事例については本誌2004年2月号，pp.30-39の記事「LSI検証クライシスの『傾向と対策』」で紹介させていただいたので説明を割愛します．テストベンチのアーキテクチャは，UMLによって可視化しています．これにより，検証チーム内の共通理解を促進しただけでなく，設計者とのコミュニケーションも円滑になりました．オブジェクト指向分析を行ってアクタ（役割）を明らかにしたため，テストベンチの並行開発も行えました．また，構成要素や共通要素を責務ごとに部品化したことにより，再利用性や保守性が高まりました．

● まず，仕様や検証内容をUMLで整理してみよう

みなさんも業務の中でだれかに何かを伝えるとき，文字だけでは伝わらないので，図を示して説明することはたびたびあると思います．UMLも同じです．UMLは表記法を統一しているため，表記記号が何を表しているかの説明を行う手間が省けます．また，図のとらえかたが人によって異なる影響を最小限に抑えられます．実際，ハードウェアの知識が乏しい情報系出身の技術者（本記事の筆者のひと

1

り．オブジェクト指向とUMLの知識は持っている）でも，簡単にテストベンチを開発できました．最近のソフトウェア技術者はUMLの知識を持っている人が多いため，検証エンジニアがハードウェアの検証内容をUMLで表現して説明すれば，きちんと理解してくれると思います．そうなれば，ハードウェアとソフトウェアの間の設計文化の障壁を取り除けると期待できます．

とは言え，いきなりチーム内であすからUMLを導入することは難しいと思います．まず，自分でUMLを使って仕様や検証内容を整理してみてはいかがでしょうか．自分の中で実践するだけでもUMLのメリットを享受でき，UMLの必要性を実感できると思います．

なお，筆者らが参加しているUMTP（UMLモデリング推進協議会）SoC分科会（http://www.umtp-japan.org/activity/07.html）では，ASIC開発やシステムLSI開発における要求分析やモデリングのノウハウをナレッジ化し，LSI開発における最適なモデリング手法は何かを極めることを目的として活動しています．興味のある方は，ぜひご参加ください．

参考・引用*文献

(1) 渡辺博之，芳村美紀，桑本茂樹，敷山喜与彦；思考系UMLモデリング即効エクササイズ，翔泳社，2004年.

みやはら・ただよし
はぎわら・あつし
もりた・なおや
（株）リコー MFP事業本部 GW開発センター

<筆者プロフィール>

宮原忠義．1986年にリコーに入社．ASICの回路設計および検証に従事しつつ，ASIC開発プロセス改革を目指す．イノベータであり続けることをモットーに，上流設計検証への改革に奮闘している．
萩原 篤．2001年にリコーに入社．ASICの検証業務に従事．新しいものが好きで，いろいろな技術に手を出しながら検証精度，効率を上げるために日夜努力している．
守田直也．2003年にリコーに入社．新人研修でオブジェクト指向開発を学び，現在はASICの検証業務に携わっている．最近は，要求を分析するのではなく，要求を開発するアプローチに注目している．

COLUMN UMLを導入する際に役立つ書籍

ここでは筆者らがUMLを導入するにあたって，参考にした書籍を紹介したいと思います．

●『独習UML 改訂版』（ジョゼフ・シュムラー 著，翔泳社 刊，2002年発行）

UML，オブジェクト指向をまんべんなく学ぶのに最適な1冊です．最新版はUML2.0の説明が多く，まだ不明確な部分があるので，改訂版から読むことをお勧めします．UMLのダイアグラムごとに説明がまとめてあるため，知りたいダイアグラムから読むこともできます．ダイアグラムの具体例もわかりやすく，初めてUMLを学ぶのによい書籍だと思います．

●『思考系UMLモデリング即効エクササイズ』（渡辺博之，芳村美紀，桑本茂樹，敷山喜与彦 著，翔泳社 刊，2004年発行）

モデリングには必要不可欠であるシステムの本質を見極める力が養える1冊です．本文でも書きましたが，筆者らはこの書籍を勉強会に使用しました．クイズ形式で記載されているため，勉強会に採用しやすいと思います．また，クイズを解くだけでなく，回答を説明し，指摘を受けることで，モデリング力を効率良く獲得できると思います．

●『組み込みUML』（渡辺博之，渡辺政彦，堀松和人，渡守武和記 著，翔泳社 刊，2002年発行）

UMLを利用してモデル化した後，設計・実装はどう行うのか，オブジェクト指向開発を組み込みシステム開発にどう適用するのか，といった問題を解決してくれる書籍です．

●『Software People vol.3』（技術評論社 刊，2003年発行）

UMLを導入するにあたって失敗しないために役立つ本です．UMLの導入に失敗する原因と成功するポイントを簡潔にまとめてあります．UMLの導入を検討されているマネージャの方には役に立つのではないでしょうか．

●『UML 500の技』（Windowsプログラミング愛好会 著，技術評論社 刊，2003年発行）

モデリングの最中に，「あれっ，この表記は何を表しているんだっけ？」，「こういうモデリングはどう書くの？」といったことを調べるのに使える1冊です．一問一答形式で記載されているため，わからないことをすぐに調べられます．筆者らは辞書のように使っていました．

＊　　＊　　＊

ここで紹介した書籍以外にも，最近は書店に行けばUMLに関する書籍が多数刊行されています．いろいろな書籍を手にとって読み比べ，みなさんにとって最適な1冊を見つけていただけたらと思います．